A robust and efficient solver based on kinetic schemes for Magnetohydrodynamics (MHD) equations.

Hubert Baty, Florence Drui, Philippe Helluy, Emmanuel Franck, Christian Klingenberg, Lukas Thanhäuser

A robust and efficient solver based on kinetic schemes for Magnetohydrodynamics (MHD) equations.

Hubert Baty[1], Florence Drui[2], Emmanuel Franck[3], Philippe Helluy[3], Christian Klingenberg[4], and Lukas Thanhäuser[4]

Abstract This paper is devoted to the simulation of magnetohydrodynamic (MHD) flows with complex structures. This kind of flow present instabilities that generate shock waves. We propose a robust and accurate numerical method based on the Lattice Boltzmann methodology. We explain how to adjust the numerical viscosity in order to obtain stable and accurate results in smooth or discontinuous parts of the flow and reduced divergence errors. This method can handle shock waves and can be made second order in smooth regions. It is also very well adapted to computing with Graphics Processing Unit (GPU). We also give results for a tilt instability test case on very fine meshes.

1 Introduction

The MagnetoHydroDynamics (MHD) system is a fundamental model used in many fields of physics: astrophysics, plasma physics, geophysics... Indeed, the MHD model is commonly adopted as an excellent framework for collisional plasma environments. The numerical approximation of this system is a difficult task. Compared to other models in fluid mechanics, it contains more conservative unknowns and thus more scales and more different wave speeds. It is also subject to complex phenomena such as occurrence of shock waves, current sheet formation, magnetic reconnection, instabilities and turbulent behaviors. An additional specificity is that the magnetic field has to satisfy a divergence-free condition, which is generally difficult to be verified by numerical solutions. In order to deal with the divergence-free condition, we adopt here a modified version of the MHD equations by a divergence cleaning term proposed in [16].

[1]Observatoire de Strasbourg · [2]CEA Saclay · [3]IRMA, Université de Strasbourg, Inria Tonus, philippe.helluy@unistra.fr · [4]University of Würzburg

We propose a simple scheme, based on an abstract kinetic interpretation, for computing two-dimensional MHD solutions. The kinetic interpretation is vectorial and has been first proposed by Bouchut in [5] and Aregba-Natalini in [1]. In this approach, the original system of conservation laws is represented by a system of transport equations coupled through a so-called collision source term. The idea originates from the Boltzmann kinetic theory of gases, but it is purely abstract and has no physical meaning. It leads to natural numerical methods, where the transport step and the collision step are made separately. Other interesting features arise from this representation (see [2, 12], for instance).

In this paper, we solve the kinetic representation with a Lattice-Boltzmann Method (LBM), where the transport step is solved exactly on a regular grid. The same approach has already been used in [21] for solving compressible Euler equations. An important aspect of the LBM is the choice of the relaxation parameter in the collision step. The choice of the parameter allows adjusting the numerical viscosity of the LBM scheme. We provide an analysis in a simplified one-dimensional framework which shows that it is possible to adjust more precisely the numerical viscosity with a generalized matrix relaxation parameter. We also show that the divergence cleaning effect is improved if the relaxation parameter is chosen differently for the physical variables and the divergence cleaning potential.

In order to capture fine structures, it is necessary to consider very fine meshes. We have programmed the algorithm in a very efficient way in order to address recent GPUs (Graphic Processing Units) or multicore CPUs. We describe the implementation, which relies on OpenCL and PyOpenCL, and the memory optimizations used for reaching high performance. The program allows performing full MHD simulations on grids as fine as 7000×7000 within a few hours.

Finally, we apply the whole approach to several MHD simulations: the classical Orszag-Tang test and a more physical study of ideal MHD instabilities (tilt modes) and associated formation of quasi-singular current sheets.

2 Mathematical model

The MagnetoHydroDynamic (MHD) system is a model used in many fields of physics. It consists of an extension of the compressible Euler equations for taking into account magnetic effects. A difficulty is that the magnetic field has to satisfy a divergence-free condition. This condition expresses that magnetic charges are not observed in Nature. Standard finite volume methods do not guarantee that the numerical magnetic field is divergence-free. More annoying: the divergence errors generally grow with the simulation time, which leads to physically wrong results. For limiting the divergence errors, we adopt the divergence cleaning method described in [16].

2.1 MHD equations with divergence cleaning

We consider the MHD equations with Divergence Cleaning [16] (called the MHD-DC equations in the following)

$$\partial_t \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ Q \\ \mathbf{B} \\ \psi \end{pmatrix} + \nabla \cdot \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u}\otimes\mathbf{u} + (p + \frac{\mathbf{B}\cdot\mathbf{B}}{2})\mathbf{I} - \mathbf{B}\otimes\mathbf{B} \\ (Q + p + \frac{\mathbf{B}\cdot\mathbf{B}}{2})\mathbf{u} - (\mathbf{B}\cdot\mathbf{u})\mathbf{B} \\ \mathbf{u}\otimes\mathbf{B} - \mathbf{B}\otimes\mathbf{u} + \psi\mathbf{I} \\ c_h^2\mathbf{B} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The velocity and magnetic field are denoted

$$\mathbf{u} = (u_1, u_2, u_3)^T, \quad \mathbf{B} = (B_1, B_2, B_3)^T,$$

the pressure is given by a perfect-gas law with a constant polytropic exponent $\gamma > 1$

$$p = (\gamma - 1)(Q - \rho\frac{\mathbf{u}\cdot\mathbf{u}}{2} - \frac{\mathbf{B}\cdot\mathbf{B}}{2}).$$

The other variables are the density ρ, the total energy Q, the divergence cleaning potential ψ. The divergence cleaning velocity is a positive parameter $c_h > 0$.

When the magnetic field satisfies the divergence-free condition

$$\nabla \cdot \mathbf{B} = 0,$$

and the potential ψ is a constant, then the MHD-DC equations simply reduce to the usual MHD equations. In other words, the MHD-DC equations are a generalization of the MHD equations, where the magnetic field can have a non-vanishing divergence.

It is possible to show that the MHD-DC system, as the MHD system, is hyperbolic [16]. However this system is not strictly hyperbolic, which leads to some difficulties, such as non-uniqueness of the Riemann problems in some situations [41].

The interest of the MHD-DC formulation is for numerical approximations. Indeed, standard approximations of the usual MHD equations suffer from drifting errors along time of the divergence constraint. This has been observed by several authors [36, 16, 3]. A review on this topic is developed for instance in [42]. Approximations of the MHD-DC generally have a much better behavior. In this generalized model, the divergence errors propagate at the wave speed c_h. The errors are then damped at the boundaries of the computational domain or by the numerical diffusion in the domain. In [16] other divergence corrections are proposed. We choose a correction that leads to a conservative and hyperbolic model, in order to be able to apply the general theory of vectorial kinetic representations given in [5, 1]. Theoretically,

the parameter c_h can take any value. But in practice it is generally chosen larger than all the wave speeds of the MHD system (see [16]).

We introduce the conservative variables

$$\mathbf{w} = \mathbf{w}(\mathbf{x}, t) = \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ Q \\ \mathbf{B} \\ \psi \end{pmatrix} \in \mathbb{R}^m, \quad m = 9.$$

and the flux ($\mathbf{n}$ is a vector of $\mathbb{R}^3$)

$$\mathbf{F}(\mathbf{w}, \mathbf{n}) = \begin{pmatrix} \rho\mathbf{u}\cdot\mathbf{n} \\ \rho\mathbf{u}\cdot\mathbf{n}\mathbf{u} + (p + \frac{\mathbf{B}\cdot\mathbf{B}}{2})\mathbf{n} - \mathbf{B}\cdot\mathbf{n}\mathbf{B} \\ (Q + p + \frac{\mathbf{B}\cdot\mathbf{B}}{2})\mathbf{u}\cdot\mathbf{n} - (\mathbf{B}\cdot\mathbf{u})\mathbf{B}\cdot\mathbf{n} \\ \mathbf{u}\cdot\mathbf{n}\mathbf{B} - \mathbf{B}\cdot\mathbf{n}\mathbf{u} + \psi\mathbf{n} \\ c_h^2\mathbf{B}\cdot\mathbf{n} \end{pmatrix} \in \mathbb{R}^m.$$

In this work, we assume that all the fields do not depend on the x_3 space variable. We are thus computing two-dimensional solutions. If we set

$$\mathbf{n}_1 = (1,0,0)^T, \quad \mathbf{n}_2 = (0,1,0)^T, \quad \mathbf{F}_1 = \mathbf{F}(\mathbf{w}, \mathbf{n}_1), \quad \mathbf{F}_2 = \mathbf{F}(\mathbf{w}, \mathbf{n}_2),$$

the MHD equations can also be written as a two-dimensional system of nine conservation laws

$$\partial_t \mathbf{w} + \partial_1 \mathbf{F}_1 + \partial_2 \mathbf{F}_2 = 0, \tag{1}$$

where we use the notation ∂_i for the partial derivative $\partial/\partial x_i$. If the fields depend only on the x_1 space variable, the system reduces to

$$\partial_t \mathbf{w} + \partial_1 \mathbf{F}_1 = 0. \tag{2}$$

In this case, the mathematical analysis is simplified. We shall perform an analysis of the numerical viscosity of the kinetic method in this simplified framework.

2.2 Kinetic representation

The Lattice-Boltzmann Method (LBM) originated from the physical kinetic interpretation of the Navier-Stokes equations [10]. In the physical world, the fluid particles can have arbitrary velocities. The main idea of the LBM is that it is possible to construct abstract kinetic interpretations of the Navier-Stokes equations in which the particles velocities can take a few number of given values. This makes it possible to solve the kinetic model directly in an efficient way. We refer to [40] for a history of the LBM.

Initially devised for solving incompressible Navier-Stokes equations, the LBM is based on a single scalar particle distribution function. More recently, it has been extended to other systems of conservation laws. Extensions to MHD can be found in [17, 32]. Dellar, in [17] showed that it is not possible to rely on a single scalar kinetic function for approximating the MHD equations. He proposes to represent the magnetic part of the equations with a vectorial kinetic formulation. Another vectorial kinetic interpretation of the MHD equations was also previously proposed in [14] for building a numerical flux for a Finite Volume method.

The resulting hybrid kinetic model (scalar for the fluid part, vectorial for the magnetic part) is, however, limited to low-Mach number flows.

Several attempts have been done to extend the LBM to arbitrary Mach flows with a scalar kinetic function. For instance, in [23, 31, 24] the Maxwell distribution is approximated with more velocities in the lattice. The additional degrees of freedom allow to computing a discrete Maxwellian distribution that matches exactly the moments of the compressible Euler equations. These approaches require quite large lattices. For instance, in [31] the authors propose a D3Q39 LBM with 39 kinetic velocities for approximating the five-equation, three-dimensional, compressible Euler equations. In addition, the methods proposed in [23, 31, 24] become unstable when the temperature exits from a given range. Another method is presented in [37] for improving the stability of the scalar LBM for compressible flow. It relies on the classical D2Q9 or D3Q27 lattices. However, the LBM is supplemented with a finite-difference approximation of the temperature equation. This approach is thus not fully an LBM method.

For addressing transonic and supersonic flows, we think that a fully vectorial kinetic model is preferable. The general theory of lattice vectorial kinetic model is discussed in [5, 1, 26, 21]. It is valid for any hyperbolic system of conservation laws and is no more limited to low-Mach number flows. It relies on a relatively small set of kinetic data. For instance, the LBM proposed by Dubois in [21] requires only 30 kinetic transported kinetic quantities compared to the 39 kinetic velocities of [31]. In this work Dubois observes that while very robust and promising, his method is quite numerically diffusive. For obtaining accurate results, it is therefore necessary to consider refined meshes.

Let us finally mention that the vectorial kinetic approach is also of interest for low Mach numbers flows and the incompressible limit [6, 43].

The vectorial kinetic approach can be used on arbitrary unstructured meshes at any order of approximation [2, 11]. When the lattice velocities are aligned with the mesh, it is possible to adopt the simple exact transport step of the LBM. This is the method that we present here.

We consider a real number $\lambda > 0$ and four velocities $\mathbf{v}_k = (v_k^1, v_k^2)^T$, $k = 1 \dots 4$, defined by

$$\mathbf{v}_1 = \begin{pmatrix} -\lambda \\ 0 \end{pmatrix}, \quad \mathbf{v}_2 = \begin{pmatrix} \lambda \\ 0 \end{pmatrix}, \quad \mathbf{v}_3 = \begin{pmatrix} 0 \\ -\lambda \end{pmatrix}, \quad \mathbf{v}_4 = \begin{pmatrix} 0 \\ \lambda \end{pmatrix}.$$

We have four vectorial distribution functions $\mathbf{f}_k(\mathbf{x},t) \in \mathbb{R}^m$, $k = 1 \dots 4$. The conservative variable $\mathbf{w}$ is related to the kinetic data by

$$\mathbf{w} = \sum_{k=1}^{4} \mathbf{f}_k.$$

Usually, in the Lattice-Boltzmann philosophy, the kinetic system is a set a transport equations coupled through a relaxation source term and reads (with $\tau > 0$)

$$\partial_t \mathbf{f}_k + \mathbf{v}_k \cdot \nabla \mathbf{f}_k = \boldsymbol{\mu}_k = \frac{1}{\tau}(\mathbf{f}_k^{eq} - \mathbf{f}_k), \quad k = 1 \dots 4. \tag{3}$$

The equilibrium function $\mathbf{f}_k^{eq}$ also called the Maxwellian state is chosen as

$$\mathbf{f}_1^{eq}(\mathbf{w}) = \frac{1}{4}\mathbf{w} - \frac{1}{2\lambda}\mathbf{F}_1(\mathbf{w}), \quad \mathbf{f}_2^{eq}(\mathbf{w}) = \frac{1}{4}\mathbf{w} + \frac{1}{2\lambda}\mathbf{F}_1(\mathbf{w}),$$

$$\mathbf{f}_3^{eq}(\mathbf{w}) = \frac{1}{4}\mathbf{w} - \frac{1}{2\lambda}\mathbf{F}_2(\mathbf{w}), \quad \mathbf{f}_4^{eq}(\mathbf{w}) = \frac{1}{4}\mathbf{w} + \frac{1}{2\lambda}\mathbf{F}_2(\mathbf{w}).$$

One can check that

$$\mathbf{w} = \sum_{k=1}^{4} \mathbf{f}_k^{eq}, \quad \mathbf{F}_i = \sum_{k=1}^{4} v_k^i \mathbf{f}_k^{eq}. \tag{4}$$

When the relaxation time $\tau \to 0$ then from (3) we see that

$$\mathbf{f}_k \simeq \mathbf{f}_k^{eq},$$

and thus from (4) we recover the MHD equations (1).

But in practice, it is quite difficult to solve equation (3) directly because the relaxation source term couples all the kinetic transport equation. It is better to replace the source term in such a way that most of the time it vanishes and thus the coupling is limited. For this, we introduce the Dirac comb Ψ defined by

$$\Psi(t) = \sum_{i \in \mathbb{Z}} \delta(t - i\Delta t),$$

where δ is the usual Dirac measure and Δt a positive time step. Let ω be a relaxation parameter $\in [1,2]$. The source term is then written as

$$\boldsymbol{\mu}_k(\mathbf{x},t) = \omega\Psi(t)\left(\mathbf{f}_k^{eq}(\mathbf{w}(\mathbf{x},t^-)) - \mathbf{f}_k(\mathbf{x},t^-)\right).$$

In other words, at times $t = i\Delta t$, $\mathbf{f}$ has jumps in time and the values after the jump

$$\mathbf{f}_k(\mathbf{x}, t^+) = \mathbf{f}_k(\mathbf{x}, t^-) + \omega \left(\mathbf{f}_k^{eq}(\mathbf{w}(\mathbf{x}, t^-)) - \mathbf{f}_k(\mathbf{x}, t^-) \right).$$

This comes from the definition of the time derivative in the weak sense. The computation of the jumps at times $t = i\Delta t$ are called the collision steps. The rest of the time, $\mathbf{f}_k$ is a solution of the free transport equation

$$\partial_t \mathbf{f}_k + \mathbf{v}_k \cdot \nabla \mathbf{f}_k = 0. \tag{5}$$

The coupling indeed occurs only at times $t = i\Delta t$.

Let us remark that when $\omega = 1$ the collision step reads

$$\mathbf{f}_k(\mathbf{x}, t^+) = \mathbf{f}_k^{eq}(\mathbf{w}(\mathbf{x}, t^-)).$$

It is a simple projection on the Maxwellian state associated to the conservative data. This scheme is a first order in time approximation of the original equations. An interesting case is the case of over-relaxation $\omega = 2$. Then the scheme is a second order in time approximation of the original equations. See for instance [18]). See also below for a mathematical analysis in the one-dimensional case.

3 Numerical method

For solving (3) numerically, we first construct a structured grid of the square $\mathcal{D} =]0, L[\times]0, L[$. The space step is given by $\Delta x = \frac{L}{N}$. and the grid-points are then as follows

$$\mathbf{x}_{i,j} = \begin{pmatrix} (i + \frac{1}{2})\Delta x \\ (j + \frac{1}{2})\Delta x \end{pmatrix}, \quad i, j \in \{0, \ldots, N-1\}.$$

For a simpler presentation we can assume periodic boundary condition, which amounts to the following equivalences

$$i + N \equiv i, \quad j + N \equiv j.$$

We denote by $\mathbf{w}_{i,j}^n$ and $\mathbf{f}_{k,i,j}^n$ the approximation of $\mathbf{w}$ and $\mathbf{f}_k$ at the grid points $\mathbf{x}_{i,j}$ and time $t_n = n\Delta t$ just after the collision step. The values of the kinetic vectors just before the collision step are denoted $\mathbf{f}_{k,i,j}^{n,-}$. For solving the kinetic system (3) we treat the transport and the relaxation terms separately.

3.1 Transport solver

The transport equation (5) admits an exact solution

$$\mathbf{f}_k(\mathbf{x}, t + \Delta t) = \mathbf{f}_k(\mathbf{x} - \Delta t \mathbf{v}_k, t).$$

If we assume that the time step satisfies $\Delta t = \frac{\Delta x}{\lambda}$, the transport operator then reduces to a simple shift. Before the collision step, we have thus

$$\begin{cases} \mathbf{f}_{1,i,j}^{n+1,-} = \mathbf{f}_{1,i+1,j}^{n}, \mathbf{f}_{2,i,j}^{n+1,-} = \mathbf{f}_{2,i-1,j}^{n}, \\ \mathbf{f}_{3,i,j}^{n+1,-} = \mathbf{f}_{3,i,j+1}^{n}, \quad \mathbf{f}_{4,i,j}^{n+1,-} = \mathbf{f}_{4,i,j-1}^{n}. \end{cases} \tag{6}$$

3.2 Relaxation

At the end of the transport step, we can compute the conservative data

$$\mathbf{w}_{i,j}^{n+1} = \sum_{k=1}^{4} \mathbf{f}_{k,i,j}^{n+1,-}. \tag{7}$$

The usual projection method would be to write

$$\mathbf{f}_{k,i,j}^{n+1} = \mathbf{f}_k^{eq}(\mathbf{w}_{i,j}^{n+1}). \tag{8}$$

As stated above, it is more accurate to consider an over-relaxation

$$\mathbf{f}_{k,i,j}^{n+1} = 2\mathbf{f}_k^{eq}(\mathbf{w}_{i,j}^{n+1}) - \mathbf{f}_{k,i,j}^{n+1,-}.$$

It is possible to add a small dissipation $\tau > 0$. See [26] or [11] for the relaxation scheme in the case $\tau > 0$. This dissipation provides a better stability in numerical cases, when sharp fronts are generated.

The relaxation step finally reads

$$\mathbf{f}_{k,i,j}^{n+1} = \omega \mathbf{f}_k^{eq}(\mathbf{w}_{i,j}^{n+1}) - (\omega - 1)\mathbf{f}_{k,i,j}^{n+1,-}, \tag{9}$$

where $\omega = \frac{2\Delta t}{2\tau + \Delta t}$. Typically, one will choose $\omega = 1.9$ in our simulations. We see that the whole algorithm is extremely simple. It is a succession of "shifts" (6) and "collisions" (9). For $\omega = 2$ the scheme is second order but unstable in shocks. For $\omega = 1$ the scheme is very robust, entropy dissipative, but quite diffusive. It would be interesting to construct a rigorous strategy for choosing locally the optimal value of ω. Let us mention that many finite volume schemes have been designed for solving MHD equations, with specific Riemann solvers. We can mention for instance the solver of [7], based on a relaxation approach, with proven stability and accuracy features. However, the solver presented in [7] is more complicated to program and less computationally efficient. We can thus compensate the slightly lower accuracy of the kinetic scheme by finer meshes, without increasing too much the computational load.

3.3 Boundary conditions

For the moment, we have assumed periodic boundary conditions. But Dirichlet or Neumann conditions are also possible. Neumann conditions are a good candidate for handling variables associated with outgoing waves. Indeed, because of the hyperbolic nature of the equations, it is not possible to impose all the physical data at the boundaries. The number of boundary conditions depends on the wave pattern. We refer to [19], for a specific analysis of the boundary conditions in the over-relaxation scheme.

3.4 Analysis of the numerical viscosity in the one-dimensional case

In this section, we state some results about the numerical viscosity of the kinetic relaxation scheme in the one-dimensional case. This one-dimensional analysis will give us simple intuitions for adjusting the relaxation parameter in the two-dimensional case. We consider the one-dimensional MHD-DC system (2). For more simplicity, we will denote $\mathbf{F} = \mathbf{F}_1$ and $x = x_1$. The equations then read

$$\partial_t \mathbf{w} + \partial_x \mathbf{F}(\mathbf{w}) = 0.$$

For the analysis, it is possible to replace the scalar relaxation parameter ω by a matrix $\boldsymbol{\Omega}$, for more generality. In the following, we establish the comparison between matrices in the usual way, by the comparison of the associated quadratic form (the resulting order is thus not total).

It is then possible to prove the following result:

Theorem 1 *If the relaxation matrix satisfies* $\mathbf{I} < \boldsymbol{\Omega} < 2\mathbf{I}$ *and if* $\mathbf{f} = \mathbf{f}^{eq}$ *at the initial time, then, up to second-order terms in* $O(\Delta t^2)$, $\mathbf{w}$ *is a solution of the following system of conservation laws*

$$\partial_t \mathbf{w} + \partial_x \mathbf{F}(\mathbf{w}) = \lambda^2 \Delta t \partial_x \left((\boldsymbol{\Omega}^{-1} - \frac{1}{2}\mathbf{I})(\mathbf{I} - \frac{1}{\lambda^2}\mathbf{F}'(\mathbf{w})^2)\partial_x \mathbf{w} \right) + O(\Delta t^2). \quad (10)$$

Remark 1 The proof is based on standard Taylor expansions. For the scalar case $\boldsymbol{\Omega} = \omega\mathbf{I}$, it can be found (for instance) in [13]. The approach is classical in the analysis of the Lattice Boltzmann Method (LBM). See for instance [20, 34].

Remark 2 For $\boldsymbol{\Omega} = \mathbf{I}$ the proof, based on a Chapman-Enskog expansion, is also given in [9].

The above analysis allows recovering formally the so-called sub-characteristic condition. Assuming that $\mathbf{I} < \boldsymbol{\Omega} < 2\mathbf{I}$, the second-order ("viscous") terms have the good sign, which ensures stability of the model, if the following matrix is positive:

$$\mathbf{V}(\lambda, \mathbf{w}) = \mathbf{I} - \frac{1}{\lambda^2}\mathbf{F}'(\mathbf{w})^2 > 0. \tag{11}$$

A fully rigorous mathematical proof of stability of the kinetic model is given by Bouchut in [5], Section 3.2, pp. 140–142. Bouchut's proof is not based on asymptotic expansions but on fully non-linear entropy estimates. Let us emphasize that the vectorial kinetic construction ensures stability even when shock waves occur and is not limited by a low-Mach assumption.

From (10), we formally observe that the scheme is second-order accurate in time in the over-relaxation case, when

$$\boldsymbol{\Omega} = 2\mathbf{I}.$$

Finally, this analysis provides a way to numerically approximate, up to second order in time, the second-order system of conservation laws

$$\partial_t \mathbf{w} + \partial_x \mathbf{F}(\mathbf{w}) - \partial_x \left(\mathbf{A}(\mathbf{w})\partial_x \mathbf{w}\right) = 0. \tag{12}$$

If the diffusion matrix $\mathbf{A}(\mathbf{w})$ is small, it is natural to take

$$\boldsymbol{\Omega} = 2\left(\mathbf{I} + \frac{2}{\lambda^2 \Delta t}\mathbf{A}(\mathbf{w})\mathbf{V}(\lambda, \mathbf{w})^{-1}\right)^{-1}. \tag{13}$$

Let us remark that if λ is large enough then the above matrix is well defined.

In order to check practically the accuracy of the approximation, we apply the above analysis for a simplified system of two conservation laws.

We consider the one-dimensional isothermal Euler equations with a diagonal diffusion of $\epsilon\partial_{xx}\left(\rho, \rho u\right)^T$. Hence the full fluid system we are interested in is given by

$$\partial_t \begin{pmatrix} \rho \\ \rho u \end{pmatrix} + \partial_x \begin{pmatrix} \rho u \\ \left(\rho u^2 + c^2 \rho\right) \end{pmatrix} = \epsilon \partial_{xx} \begin{pmatrix} \rho \\ \rho u \end{pmatrix}. \tag{14}$$

This system is a very simplified version of the MHD equation, where the magnetic field is assumed to vanish and the gas is supposed to be isothermal. We have chosen to use this specific non-physical diffusion for our first test since it is exactly the type of diffusion that is apparent in a standard finite volume code using a Lax-Friedrichs flux.

The diffusion matrix for (14) is simply given by

$$\mathbf{A}(\mathbf{w}) = \begin{pmatrix} \epsilon & 0 \\ 0 & \epsilon \end{pmatrix}. \tag{15}$$

From (13), the inverse of the relaxation matrix is given by

$$\boldsymbol{\Omega}^{-1} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \tag{16}$$

with

$$a_{1,1} = \frac{\epsilon\left(\left(\lambda^2 - c^2\right) - 3u^2\right)}{\Delta t\left(u^4 + \left(c^2 - \lambda^2\right)^2 - 2u^2\left(c^2 + \lambda^2\right)\right)} + \frac{1}{2},$$

$$a_{1,2} = \frac{2u\epsilon}{\Delta t\left(u^4 + \left(c^2 - \lambda^2\right)^2 - 2u^2\left(c^2 + \lambda^2\right)\right)},$$

$$a_{2,1} = -\frac{2u(u-c)(c+u)\epsilon}{\Delta t\left(u^4 + \left(c^2 - \lambda^2\right)^2 - 2u^2\left(c^2 + \lambda^2\right)\right)},$$

$$a_{2,2} = \frac{\epsilon\left(u^2 + \left(\lambda^2 - c^2\right)\right)}{\Delta t\left(u^4 + \left(c^2 - \lambda^2\right)^2 - 2u^2\left(c^2 + \lambda^2\right)\right)} + \frac{1}{2}.$$

For our first test of the matrix relaxation, we solve (14) comparing the Lattice Boltzmann scheme using (13) as relaxation matrix and a standard explicit centered finite volume scheme for approximating (14). In this centered scheme, the time step is taken very small in such way that the stability condition is satisfied and that the time integration error can be neglected. In other words, the equivalent PDE of both schemes is (12). Hence, given the parameters in both schemes are set to represent the same diffusion ϵ, one should get the same type of diffusion for both schemes.
To test this we take the simple case of a stationary viscous shock.
The initial data for this shock tube problem are

$$\mathrm{w}_l = (\rho_l, \rho_l u_l)^T \quad \mathrm{w}_r = \left(\frac{\rho_l u_l^2}{c}, \frac{c}{u_l}\right)^T \tag{17}$$

where the sound speed c is set as $c = 1$ while for the left state we have chosen

$$\mathrm{w}_l = \left(2, \frac{3}{2}\right)^T. \tag{18}$$

For the diffusion we use $\epsilon = 0.1$. For the Lattice Boltzmann scheme, we set $\lambda = 40$. The results on a grid of 128 cells at time $T = 0.1$ are shown in Figure 1. The test shows that the matrix relaxation lattice Boltzmann scheme provides the correct diffusion and therefore results in the right viscous profile. The conclusion of this section is that the relaxation parameter is related to numerical viscosity. In 1D it can indeed be adjusted to fit the exact viscosity up to second order. For two-dimensional computations the analysis is more complicated, but heuristically we expect a similar behavior.

In the following, for simplicity reasons, we only consider scalar relaxation.

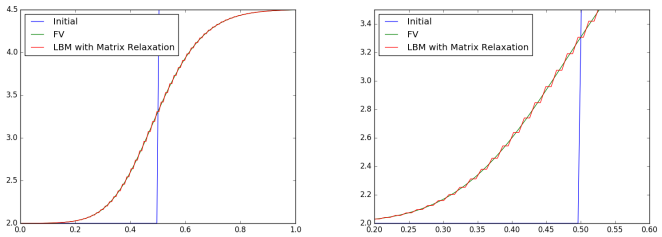

Fig. 1 Viscous Shock Test with $\epsilon = 0.1$: Comparison of the viscous profiles of the finite volume and the lattice Boltzmann scheme.

4 GPU implementation

We have implemented the above LBM algorithm. The LBM is particularly well adapted to parallelism. It is possible to provide very efficient implementations on GPU (Graphic Purpose Units) hardware.

4.1 OpenCL

4.1.1 Terminology

Today Graphic Processing Units (GPU) have more and more computation power. The Open Computing Language (OpenCL) is a software environment for simplifying the use of the GPUs for general computing. It is also possible to use OpenCL for driving a heterogeneous set of general multicore processors.

For giving an idea on how OpenCL is used in practice, we use the following terminology. An *accelerator* is a parallel computing device, such as a GPU or a multicore CPU. The *host* is the computer into which the accelerator is plugged. A *kernel* is a (generally small) program that is executed on several of the computing cores of the GPU. For instance, the Nvidia GPU GTX 1660 has 448 computing cores. Thanks to the OpenCL command queue management, it is possible to launch several million kernel instances, which are dispatched on the hundreds of cores of the GPU.

OpenCL means “Open Computing Language”.

The *OpenCL runtime* is a library of C functions, called from the host, in order to drive the GPU. The OpenCL runtime, because it is written in C is quite heavy to use in practice: the verbosity is high, the API is not very user-friendly and memory management is cumbersome. For this reason it is advised to use OpenCL wrappers written in a higher-level language such as C++ or Python. We have used the Python OpenCL wrapper written by

Andreas Klöckner, PyOpenCL [29], which makes OpenCL initializations and calls much easier and shorter to program.

The *OpenCL language* is a C-like language for writing the kernels that will be executed on the computing cores.

OpenCL is practically available since September 2009. The specification is managed by the Khronos Group (that also drives the OpenGL specification). Several books describe today OpenCL in detail. We can refer, for instance, to [25].

4.1.2 GPU

Very schematically we can consider that a general computing accelerator is made of *global memory* (typically 16 GB for the Nvidia Tesla V100) and *compute units* (typically 80).

Each compute unit is made of *processing elements* (typically 64), also called *processors.* A compute unit has a reserved *local memory* (typically 64 kb) shared by the processors of the unit.

A schematic picture of an abstract OpenCL accelerator is given in Figure 2

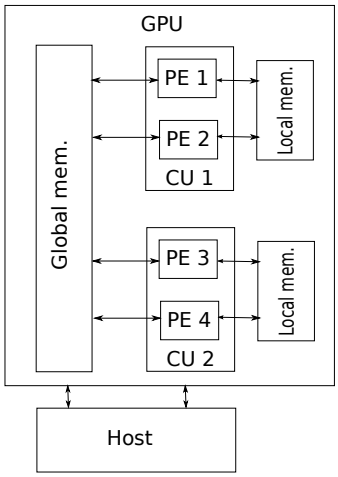

Fig. 2 A (virtual) GPU with 2 Compute Units and 4 Processing Elements

4.1.3 Programming rules

The same program can be executed on all the processing elements at the same time with the following rules:

1. All the processing elements have access to the global memory;
2. The processing elements have only access to the local memory of their compute unit;
3. The access to the global memory is relatively slow while the access to the local memory is very fast.
4. If possible, it is advised that the processing elements of the same compute unit access neighbour global memory locations, in order to improve "coalescence" (faster read/write access).
5. The memory transfers between the host memory and the GPU are slow and should be avoided;
6. If several processing elements try a read access at the same memory location (global or local) at the same time, all the reads will be successful;
7. If several processing elements try a write access at the same memory location (global or local) at the same time, ***only one write will be successful***. For some hardware, atomic operations maybe available, but should be avoided for performance reasons.

In order to perform a complex task, a kernel has to be executed many times. Each execution of a kernel is called a *work-item*. A *work-group* is a collection of work-items running on the processing elements of a given compute unit. They can access the local memory of their compute unit. Each work-item is identified by a unique global ID p.

For more details on OpenCL, we refer for instance to [25, 29].

4.2 OpenCL implementation of the LBM algorithm

We have implemented the above algorithm using PyOpenCL.

The different devices used in this paper are listed in Table 1. The AMD processor was used inside a virtual environment, which implies a non-negligible loss of performance. PyOpenCL allows us to select either the CPU or the GPU for the computations. With the OpenCL AMD drivers, when the CPU is selected, it is also possible to choose the number of activated CPU cores through a Linux environment variable. This is useful for estimating (in a crude way) the efficiency of the OpenCL parallelism.

The LBM is very simple. Our implementation is made of two OpenCL kernels, a few C functions and a small Python driver for initializing the memory buffers, launching the OpenCL kernel and plotting the results. The role of the first OpenCL kernel is to compute the initial condition directly into the memory buffer created on the OpenCL accelerator. The role of the

short name	name	type	frequency	memory	cache	CU	PE
V100	Tesla V100-PCIE-16GB	GPU	1.4 GHz	16 GB	48 kB	80	5120
Quadro	Nvidia Quadro P6000	GPU	1.5 GHz	24 GB	32 kB	30	3840
GTX	Nvidia GTX 1660	GPU	1.6 GHz	6 GB	48 kB	22	1408
AMD	AMD EPYC 7551 32-core	CPU	2 GHz	48 GB	32 kB	1-24	1-24
Intel	2 x Intel Xeon CPU E5-2609 v4	CPU	1.7 GHz	64 GB	32 kB	16	16
Iris 640	Intel Iris Graphics 640	GPU	1.0 GHz	4 GB	64 kB	48	192

Table 1 Characteristics of the OpenCL devices tested in this paper. CU stands for "Compute Units" and "PE" for "Processing Elements".

second kernel is to perform a time step of the LBM. The time-stepping is driven from the PyOpenCL program.

The most important point is to take care of the organizations of the kinetic data into memory. Indeed ensuring coalescent memory access is essential for performance. In practice all the values $\mathbf{f}_{k,i,j}$ for kinetic velocities $\mathbf{v}_k$ in cells (i,j) are arranged in a single memory buffer `fn[]`, with the following storage

$$\mathbf{f}_{k,i,j} = \texttt{fn[imem]}, \quad \texttt{imem=i+j*Nx+k*Nx*Ny},$$

where `Nx` is the number of grid points in the x_1-direction and `Ny` is the number of grid points in the x_2-direction. Then, in the OpenCL kernel, each cell (i,j) is associated to the work-item

$$p = \texttt{i+j*Nx}.$$

This numbering ensures that neighboring work-items will access neighboring memory locations during the shift algorithm (6). For instance, if work-item p access data in global memory, for reading or writing, at location `imem`, then work-item $p' = p + 1$ access location `imem+1`.

This property is still true in the relaxation algorithm. First, the kinetic data are copied in processor registers in a coalescent way. The computations of the conservative data (7), equilibrium (8) and collisions (9) are done in registers, which ensure very fast memory access. Finally, the kinetic data are copied back to the global buffer `fn[]` in a fully coalescent way.

We remark that thanks to the chosen organization into memory, we do not have to use the local cache memory for accelerating the algorithm.

In order to measure the efficiency of the implementation, we perform a memory bandwidth test for a 512×512 grid. One time-step of the method implies the read access in the global memory of the set of fields of the previous time-step. The local computations are done in registers. Then there is another write access to global memory for storing the data of the next time-step. The memory size in Gigabyte of one set of fields is

$$n_{\mathrm{GB}} = \frac{\texttt{Nx}\times\texttt{Ny}\times prec \times 4 \times m}{1024^3},$$

where $prec$ is the number of bytes for storing one floating point number ($prec = 4$ for single precision and $prec = 8$ for double precision). We then perform a given number of time iterations n_{iter} and measure the elapsed time t_{elapsed} (in s) in the OpenCL kernels. We perform two kinds of experiments. In the first experiment, we deactivate the numerical computations and only perform the shift operations. The memory bandwidth (in GB/s) of the shift algorithm is then given by

$$b = \frac{2 \times n_{\text{GB}} \times n_{\text{iter}}}{t_{\text{elapsed}}}.$$

In the second experiment, we reactivate the computations and measure how the bandwidth is reduced. This allows to evaluating how the elapsed time is shared between memory transfers and computations. The results are given in Table 2. We observe a good efficiency of the shift algorithm in the shift-only case: the transfer rates are not very far from the maximal bandwidth of the device, at least for the GPU accelerators. From these results we also observe that the LBM algorithm is clearly memory bound. When the single precision computations are activated on the GPU devices (GTX, Quadro, V100), the elapsed time of the shift-and-relaxation test is not very different from the shift-only test. For the double precision computations, we observe that the V100 device outperforms all the other GPUs.

	prec.	b (GB/s, shift-only)	b (GB/s, shift-relax)	max. theoretical b (GB/s)
Intel	float32	17.58	13.38	60
Intel	float64	19.12	17.48	60
Iris 640	float32	26.20	24.98	34
Iris 640	float64	20.08	3.78	34
GTX	float32	147.54	146.94	192
GTX	float64	148.76	49.72	192
Quadro	float32	336.45	329.06	432
Quadro	float64	344.50	127.21	432
V100	float32	692.31	676.44	900
V100	float64	705.88	610.17	900

Table 2 Bandwidth efficiency of the LBM algorithm. Comparison of the data transfer rates of the shift-only algorithm and of the shift-and-relaxation algorithm. The resulting bandwidth is compared with the maximal memory bandwidth advertised by the vendors of the hardware devices.

5 Numerical applications to MHD

5.1 Smooth vortex (performance test)

The smooth vortex test is a classical test for MHD codes. It is described for instance in [22]. Because this is an exact solution, it allows us to assess the accuracy of the solver. Here we also used this test to evaluate the efficiency of the parallel implementation. The test case is built upon a single vortex, which is a stationary solution of the MHD system, to which a constant drift velocity is added. In the moving frame centered on $\mathbf{r}_O(t) = t\mathbf{u}_{\text{drift}}$, with $\mathbf{u}_{\text{drift}} \in \mathbb{R}^2$, the analytical solution reads in polar coordinates

$$\begin{aligned}
\rho(r,\theta) &= \rho_0, \\
\mathbf{u}(r,\theta) &= u_0[\mathbf{u}_{\text{drift}} + h(r)\mathbf{e}_\theta], \\
\mathbf{B}(r,\theta) &= b_0 h(r)\mathbf{e}_\theta, \\
p(r,\theta) &= p_0 + \frac{b_0^2}{2}(1-h(r)),
\end{aligned}$$

with $b_0 = \rho_0 u_0^2$. The results shown below are obtained with $\gamma = 5/3$ and the parameter set

$$\rho_0 = p_0 = 1,\ u_0 = b_0 = 0.2,\ \mathbf{u}_{\text{drift}} = (1,1)^T,\ h(r) = \exp[(1-r^2)/2].$$

The computational domain is the square $\Omega =]-L/2, L/2[\times]L/2, L2[$, with $L = 20$. We compute the solution at time $t = 10$. The grid contains N points in x_1 and x_2 directions. For this smooth test case, we can take a relaxation parameter $\omega = 2$. We compute the error e_N in the L^1 norm at the final time between the exact solution and the numerical solution on the first component of the momentum (the other components of the solution would give similar results):

$$e_N = \int_\Omega \left|(\rho u_1)_{\text{num}} - (\rho u_1)_{\text{exact}}\right|.$$

Asymptotically, the order of the scheme is evaluated by

$$\beta \simeq \frac{\ln(e_N/e_{2N})}{\ln 2}.$$

The obtained numerical results are summed up in Table 3, where we give the convergence study and a performance evaluation of the implementation. OpenCL permits to run the same code on a multicore CPU or a GPU. We have tested several CPU or GPU hardware in single or double precision. Table 3 confirms the order of accuracy of the scheme in the case $\omega = 2$. In addition we observe a good efficiency of the implementation on several types of GPU.

On CPU there is also a speedup achieved by the OpenCL parallelism, but it is very sensitive to the OpenCL drivers. For instance, with the same hardware (an Intel Xeon two-CPU system) the program runs almost three times faster with the OpenCL Intel drivers than with the open source POCL drivers. Let us finally mention that when it is run on only one core, the code is very slow. This is due to the fact that our implementation is not really optimized for correctly harnessing the CPU cache. Here, with a more clever tiling strategy, the one-core run could probably be accelerated by an order of magnitude (see for instance [27]).

	CU	precision	N=128	N=256	N=512	N=1024	"efficiency"
AMD	1	float32	11.9 s	159 s	621 s	6396 s	1
AMD	24	float32	1.01 s	9.4 s	153 s	1380 s	5
Intel (pocl)	16	float32	2.30 s	17.3 s	96.6 s	644 s	10
Intel	16	float32	0.75 s	3.93 s	32 s	226 s	30
Intel	16	float64	0.82 s	5.62 s	53 s	315 s	20
GTX	22	float32	0.04 s	0.31s	2.46 s	19.48 s	330
Quadro	30	float32	0.017 s	0.15 s	1.06 s	8.25 s	780
Quadro	30	float64	0.15 s	0.81 s	5.67 s	45.53 s	140
V100	80	float32	0.015 s	0.084 s	0.54 s	3.93 s	1600
V100	80	float64	0.031 s	0.21 s	1.17 s	8.35 s	770
e_N		float64	0.05067625	0.013039866	0.003265470	0.00081652	
β		float64	-	1.96	1.99	2.00	

Table 3 Convergence and performance study. Some tests are done in single precision (float32) and others in double precision (float64). The "efficiency" is a comparison for N=1024 with the slowest device. "CU" means "Compute Units": it is the number of activated cores in a CPU computation or of OpenCL compute units for a GPU computation.

5.2 Orszag-Tang vortex

The Orszag-Tang test case [33, 15, 35] is often used to test a numerical method for MHD. It consists in a vortex system where turbulent structures and shocks develop. The domain is $\mathcal{D} =]0, 2\pi[\times]0, 2\pi[$ and the boundary conditions are periodic in $x_1 = x$ and $x_2 = y$. The initial conditions are given in Table 4.

In Figures 3 and 4 we present several snapshots of the evolution of the vortex. In Figure 5 we compare the results obtained with several grid refinements. The grid refinement clearly improves the sharpness of the shock profiles.

Variables	States
γ	$5/3$
ρ	γ^2
p	γ
u_x	$-\sin(y)$
u_y	$\sin(x)$
u_z	0
B_x	$-\sin(y)$
B_y	$\sin(2x)$
B_z	0

Table 4 Initial states for the Orszag-Tang test case

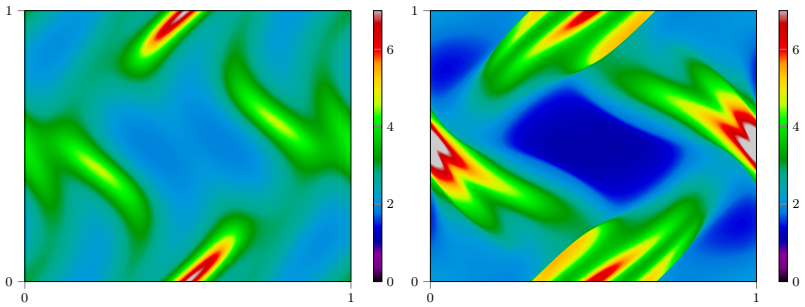

Fig. 3 Snapshots of ρ for the Orszag-Tang configuration recorded at times $t = 0.1$ and $t = 0.2$. Grid size is $Nx \times Ny = 1024 \times 1024$.

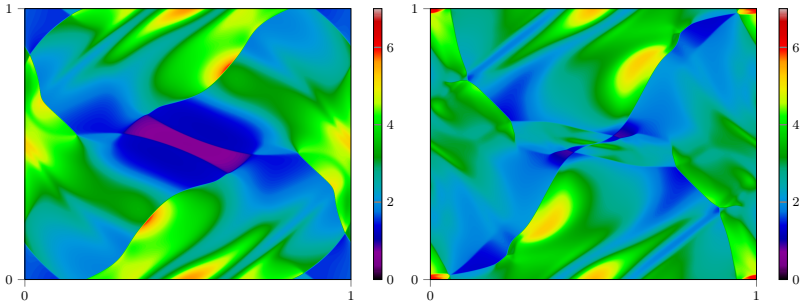

Fig. 4 Snapshots of ρ for the Orszag-Tang configuration recorded at times $t = 0.3$ and $t = 0.4$. Grid size is $Nx \times Ny = 1024 \times 1024$.

5.3 Tilt instability

The tilt instability has been studied in [38, 39, 30]. The initial magnetic field configuration for tilt instability is a dipole current structure. It consists of two oppositely directed currents embedded in a uniform magnetic field (at large distance).

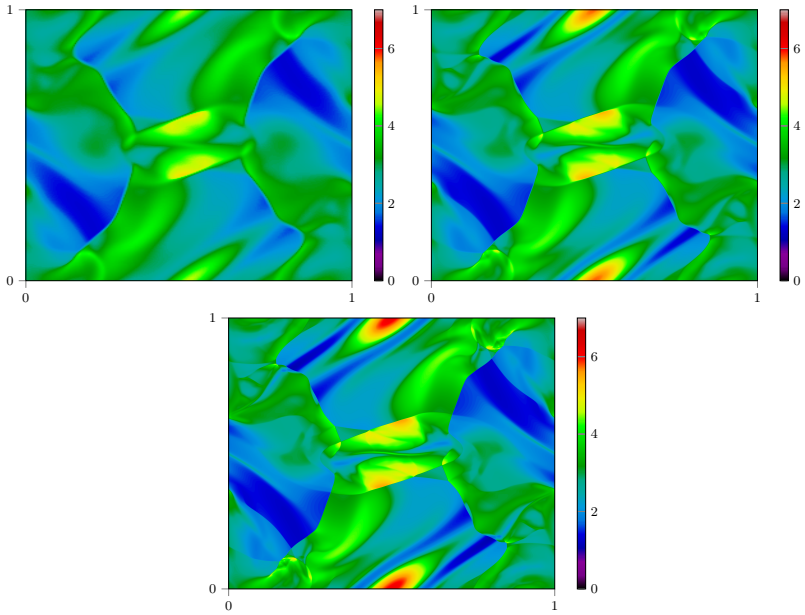

Fig. 5 Snapshots of ρ for the Orszag-Tang configuration recorded at time $t = 0.5$ and $t = 0.2$. Grid sizes are $Nx \times Ny = 256 \times 256$, 1024×1024 and 4096×4096.

These islands may be kept in an unstable equilibrium through a strong magnetic field. However, when perturbed, an instability develops that leads the islands to be aligned horizontally and to be expelled away. The tilt instability is illustrated for a simulation on a 1024×1024 grid in Figures 6 to 8 for different times and the numerical approach presented above. A detailed description of the different stages of this instability can be found in [38, 4]. In [38, 28], the study of the tilt instability was also made for a compressible MHD system, while in [39, 30] the equations are incompressible, leading to slightly different results.

In the following, we will consider a squared spatial domain of dimensions $\mathcal{D} =]-3, 3[\times]-3, 3[$ and $r^2 = x^2+y^2$. The initial condition for the equilibrium is given by:

$$\rho = 1.0, \quad \mathbf{u} = \mathbf{0}, \quad p_0 = 1.0, \quad \psi = 0, \tag{19}$$

$$k = 3.8317059702, \quad K = \frac{-2}{kJ_0(k)}, \quad \varphi = KJ_1(kr)\frac{y}{r}, \tag{20}$$

$$p = \begin{cases} p_0 & \text{if} \quad r \geq 1, \\ p_0 + \frac{k^2}{2}\varphi^2 & \text{else}, \end{cases} \tag{21}$$

$$B_1 = \begin{cases} \frac{x^2-y^2}{r^4} - 1 & \text{if} \quad r \geq 1, \\ K\left[\frac{ky^2}{r^2}J_0(kr) + \frac{x^2-y^2}{r^3}J_1(kr)\right] & \text{else}, \end{cases} \tag{22}$$

$$B_2 = \begin{cases} \frac{2xy}{r^4} & \text{if} \quad r \geq 1, \\ -K\left[\frac{kxy}{r^2}J_0(kr) - \frac{2xy}{r^3}J_1(kr)\right] & \text{else}, \end{cases} \tag{23}$$

where J_0 and J_1 are the Bessel functions of the first kind of orders zero and one respectively. A perturbation of this equilibrium initially reads:

$$\mathbf{u} = 2\epsilon \exp\left(-r^2\right) \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix},$$

with $\epsilon = 10^{-3}$.

The asymptotic magnetic field strength for large r is unity, and thus defines our normalization. We also point out that we consider non-dimensioned equations in which the magnetic permeability is set to one. Consequently, our unit time is defined as the Alfven transit time across the unit distance (i.e. the initial characteristic length scale of the dipole structure).

Moreover, for the analysis of the simulation results, we recall that the current density is given by:

$$\mathbf{C} = \nabla \times \mathbf{B},$$

and that we are interested in the third component of $\mathbf{C}$, *i.e.* C_3. We will also look at the kinetic energy of the system, which is computed as the integral over the spatial domain of the kinetic energy density:

$$E_{\text{kin}} = \int_{\mathcal{D}} \frac{1}{2}\rho|\mathbf{u}|^2.$$

We apply Dirichlet boundary conditions in this test case: the initial data in the boundary cells are simply kept constant. Finally, the numerical scheme involves the following parameters:

$$\Delta x = L_x/N_x, \quad c_h = 6 \quad \lambda = 20, \quad \Delta t = \Delta x/\lambda,$$

and $\omega = 1.9$ for all variables except from the kinetic variables associated with ψ where $\omega = 1$ (see section 5.3). In our numerical tests, we have noticed that choosing $\omega = 2$ leads to an unstable numerical solution, due to the lack of dissipation.

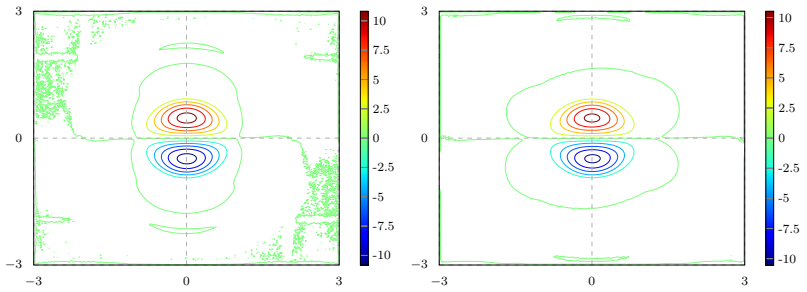

Fig. 6 Snapshots of the magnetic current density C_3 recorded at (non-dimensioned) times $t = 1$ and $t = 2$. Grid size is $Nx \times Ny = 1024 \times 1024$.

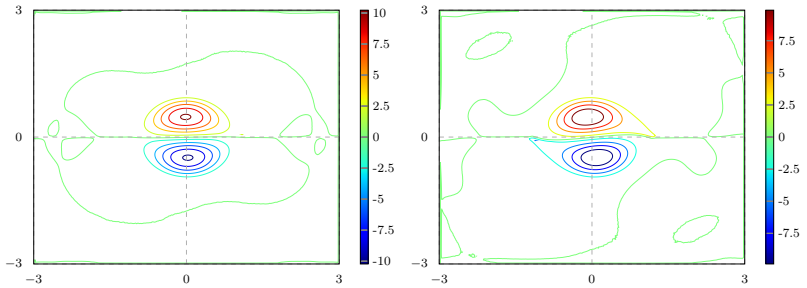

Fig. 7 Snapshots of the magnetic current density C_3 recorded at times $t = 3$ and $t = 4$. Grid size is $Nx \times Ny = 1024 \times 1024$.

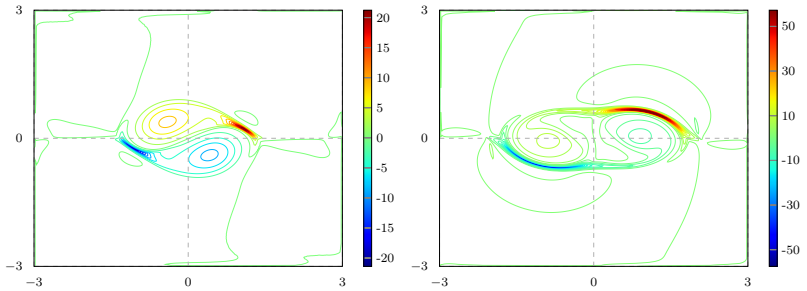

Fig. 8 Snapshots of the magnetic current density C_3 recorded at times $t = 5$ and $t = 6$. Grid size is $Nx \times Ny = 1024 \times 1024$.

Current sheets and current peak

Current sheets develop at the edges of the magnetic islands. The width of these current sheets as well as the maximum intensity of the current density depends on the numerical implementation, as studied in [30], where specific adaptive resolution (adaptive grid and order of resolution) was performed near strong current gradients. In Figure 9, we show an enlargement of the

part of the spatial domain that lies around the current sheet lying right side for simulations with different grid sizes and at time $t = 6$. The finest current structures are obtained with the finest grid (here the 4096×4096 grid). In Figure 10, we present the ratio of the maximum current density $C_{3,max}$ over the initial current density $C_{3,0}$. This ratio increases with the grid refinement and should be infinite in the case of a totally ideal MHD system ([30]). The maximum ratio we can obtain is 14.2 with the 7000×7000 grid. In comparison, [30] obtained a ratio of 5 for the simulation with $32,768$ triangles and first-order method, and a ratio of 41 for adaptive grids near the current sheets. In conclusion, our method seems to have a higher numerical resistivity, which limits the maximum current peak, but the computations are probably faster. Indeed, adaptive methods are known to suffer from overhead due to the algorithm complexity.

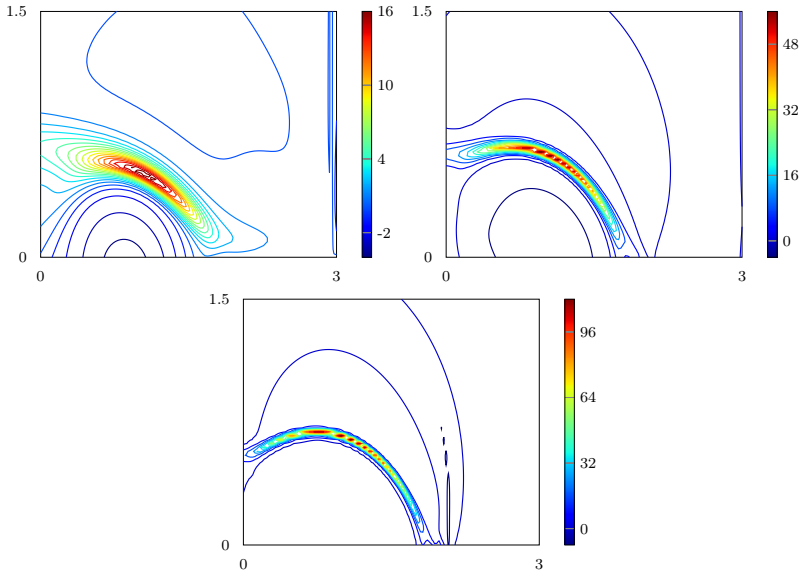

Fig. 9 Snapshots of the magnetic current density recorded at time $t = 6$ in a spatial zone around the current sheets. Grid sizes are $Nx \times Ny = 256 \times 256$, $Nx \times Ny = 1024 \times 1024$, $Nx \times Ny = 4096 \times 4096$.

Kinetic energy growth rate

During the development of the instability, the kinetic energy of the system increases exponentially. The growth rate of the kinetic energy has been studied in [38] and [30], where different values have been found. In [38], two simulations with two values of β (the ratio between the hydrodynamic and the

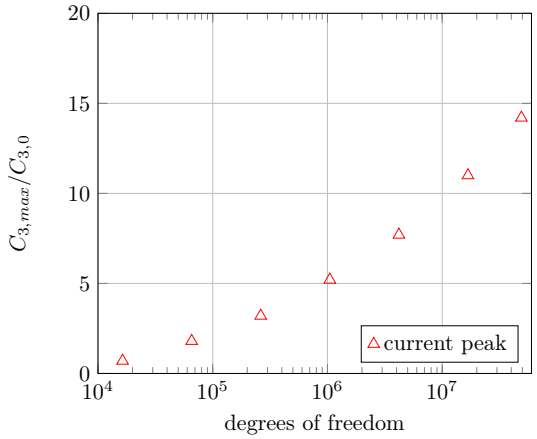

Fig. 10 Ratio of the peak current and the initial current density for simulation grids ranging from 128×128 to 7000×7000.

magnetic pressures) give growth rate values of 1.44 for low β case and of 1.27 for high β case. In [30], the study is focused on the convergence of the kinetic energy growth rate according to the grid and the order of the numerical method. The converged value for the growth rate is near 1.3 for an initial perturbation $\epsilon = 1.0 \cdot 10^{-3}$.

The simulations of the tilt instability that we perform are post-processed at regular time intervals. In Figures 11 and 12, we show the time evolution of the total kinetic energy $E_{\rm kin}(t)$. One can note that the growth of the kinetic energy happens in two stages: in a first stage, the growth is faster than exponentially, then slows down and after two seconds the second stage begins with exponential growth. When the grid resolution is higher, the first stage tends to vanish and the second stage begins sooner.

In a logarithmic scale, the linear regression of the kinetic energy growth is performed with the `Scipy` function `linregress` and the regression line is illustrated in Figures 11 and 12. The results of the linear regression are reported in Table 5 and Figure 12. The converged growth rate is close to 1.45 for our simulations. It is higher than the results of [38] and [30]. In [28] for a similar configuration, the growth rate is evaluated to 1.498. This is probably very close to the exact rate because the simulation is conducted with a very accurate adaptive scheme. Our results seem thus to be quite correct on the finest mesh.

In Table 5, we summarize the characteristics of the linear regression for each mesh size.

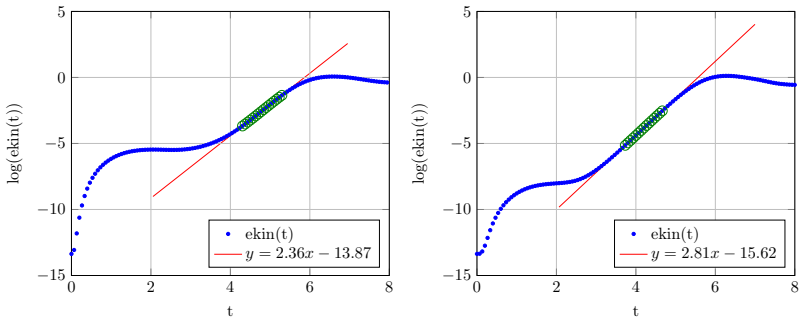

Fig. 11 Growth of the kinetic energy during the simulation. Measures that have been accounted for the linear regression are identified with green marks and the regression line is in red. Results for grid sizes $Nx \times Ny = 512 \times 512$ and $Nx \times Ny = 2048 \times 2048$.

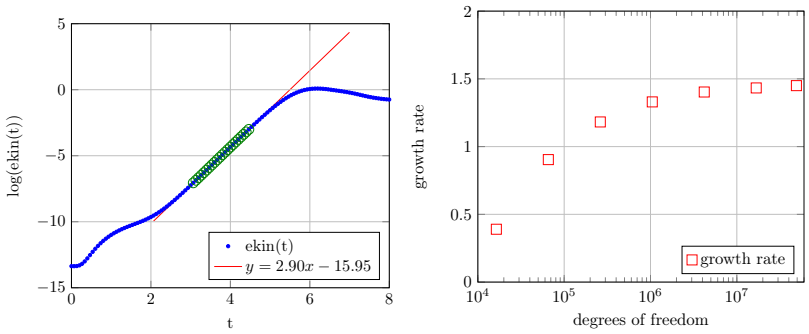

Fig. 12 Growth of the kinetic energy during the simulation. Measures that have been accounted for the linear regression are identified with green marks and the regression line is in red. Results for grid size $Nx \times Ny = 7000 \times 7000$. Growth rates from $Nx = Ny = 128$ (16384 degrees of freedom) to $Nx = Ny = 7000$ (49×10^6 degrees of freedom).

Mesh size	128^2	256^2	512^2	1024^2	2048^2	4096^2	7000^2
Regression range (s)	$[5.6, 6.7]$	$[4.9, 5.9]$	$[4.3, 5.3]$	$[3.9, 4.9]$	$[3.7, 4.7]$	$[3.0, 4.5]$	$[3.0, 4.5]$
Growth rate	0.390	0.904	1.082	1.330	1.403	1.433	1.450

Table 5 Characteristics of the linear regression for the kinetic energy growth rate for each mesh size: range of values that have been accounted for and growth rate.

Divergence cleaning effect

In order to ensure a magnetic field which is close to a divergence-free field, we have presented the divergence cleaning procedure in Section 2.1. Here we compare two strategies for the numerical implementation of the divergence cleaning equation:

1. the kinetic variables associated with the conservative variable ψ are solved numerically like any other kinetic variable. This means that in the relax-

ation step, the relaxation coefficient ω is set to $\omega = 1.9$ and the order of resolution is close to 2,

2. during the relaxation step, the kinetic variables associated with the macro-variable ψ are relaxed with a coefficient $\omega = 1$, which comes down to setting

$$\begin{aligned}\mathbf{f}_{k,i,j}^{n+1} &= \omega \mathbf{f}_k^{eq}(\mathbf{w}_{i,j}^{n+1}) - (\omega - 1)\mathbf{f}_{k,i,j}^{n+1,-} \\ &= \mathbf{f}_k^{eq}(\mathbf{w}_{i,j}^{n+1}).\end{aligned}$$

Now, the order of resolution is 1 and the waves associated with the perturbations of the divergence-free constraint are better damped than with $\omega = 1.9$. In [16], several strategies are proposed for damping the divergence errors. One of the methods is to introduce a viscous damping term. In its spirit our proposal is similar, but the viscous term is introduced here in a numerical way.

These two options are compared in Figures 13 to 15 for different times. For obtaining those plots, the divergence of the magnetic field is computed with a simple centered finite difference approximation

$$\nabla \cdot \mathbf{B} \simeq \frac{B_1(x + \Delta x, y) - B_1(x - \Delta x, y)}{2\Delta x} + \frac{B_2(x, y + \Delta y) - B_2(x, y - \Delta y)}{2\Delta y}.$$

At time $t = 0.5$, one can notice the effects of the boundary conditions that have propagated towards the center of the domain. The perturbations of the divergence-free constraint have been more attenuated with the second strategy than with the first one. At time $t = 1$, the divergence-free constraint is mainly violated at the edges of the magnetic vortices and close to the domain boundaries. Once more, the results are better with the second divergence cleaning strategy. Finally, at time $t = 5$, when the vortices have begun to align, the divergence-free constraint is strongly perturbed, but results are still better with the second strategy.

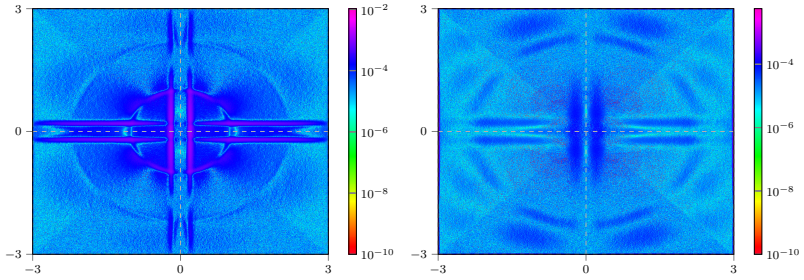

Fig. 13 Snapshots of the divergence of the magnetic field recorded at times $t = 0.5$ for both strategies of divergence cleaning. Grid size is $Nx \times Ny = 1024 \times 1024$.

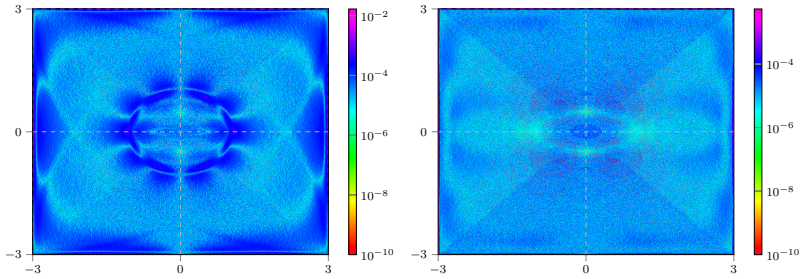

Fig. 14 Snapshots of the divergence of the magnetic field recorded at times $t = 1$ for both strategies of divergence cleaning. Grid size is $Nx \times Ny = 1024 \times 1024$.

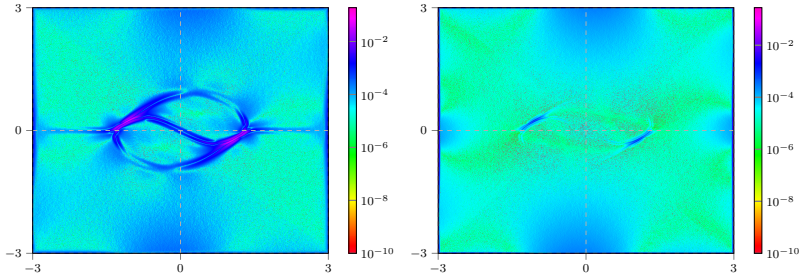

Fig. 15 Snapshots of the divergence of the magnetic field recorded at times $t = 5$ for both strategies of divergence cleaning. Grid size is $Nx \times Ny = 1024 \times 1024$.

6 Conclusion

In this work we have proposed a fast and robust Lattice-Boltzmann solver for the two-dimensional MHD equations with divergence cleaning. The method is general and can be extended to other systems of conservation laws.

We have provided a preliminary one-dimensional analysis in order to evaluate the numerical viscosity of the numerical scheme. We have shown that in principle, the numerical viscosity can be adjusted to a specific viscosity. An interesting challenge would be to extend the analysis to higher dimensions, but this leads to complex calculations because of cross second-order derivatives. If this analysis is possible, this would lead to a scheme where the physical resistive terms could be approximated properly.

The kinetic representation of the equations is very well adapted to massive parallel computing. The simplicity of the algorithm allows achieving almost optimal efficiency on GPU hardware. This leads to the possibility to conduct computations on very fine uniformly refined meshes.

When conducting the simulations, we observed that the limiting factor was memory rather than computation time. In order to extend the method to even finer meshes or to three-dimensional computations, the memory management has to be improved. An obvious possibility is to distribute the computations

on several GPUs, using a task-based runtime system (such as in [8]). This approach could also be mixed with a compression strategy in order to reduce memory occupation and data transfers.

Thanks: this work has been finalized during a one-week stay at the Oberwolfach Research Institute for Mathematics (MFO) in September 2020. The authors are very grateful to the staff of the MFO for their warm welcome and perfect organization despite the constraints caused by the pandemic.

The authors also thank Matthieu Boileau for his helpful contribution in the design of the PyOpenCL GPU code.

References

1. D. Aregba-Driollet and R. Natalini. Discrete kinetic schemes for multidimensional systems of conservation laws. *SIAM Journal on Numerical Analysis*, 37(6):1973–2004, 2000.
2. J. Badwaik, M. Boileau, D. Coulette, E. Franck, P. Helluy, C. Klingenberg, L. Mendoza, and H. Oberlin. Task-based parallelization of an implicit kinetic scheme. *ESAIM: Proceedings and Surveys*, 63:60–77, 2018.
3. T. Barth. On the role of involutions in the Discontinuous Galerkin discretization of Maxwell and magnetohydrodynamic systems. In *Compatible spatial discretizations*, pages 69–88. Springer, 2006.
4. H. Baty. FINMHD: An adaptive finite-element code for magnetic reconnection and formation of plasmoid chains in magnetohydrodynamics. *The Astrophysical Journal Supplement Series*, 243(2):23, 2019.
5. F. Bouchut. Construction of BGK models with a family of kinetic entropies for a given system of conservation laws. *Journal of Statistical Physics*, 95(1-2):113–170, 1999.
6. F. Bouchut, Y. Jobic, R. Natalini, R. Occelli, and V. Pavan. Second-order entropy satisfying BGK-FVS schemes for incompressible Navier-Stokes equations. *The SMAI journal of computational mathematics*, 4:1–56, 2018.
7. F. Bouchut, C. Klingenberg, and K. Waagan. A multiwave approximate Riemann solver for ideal MHD based on relaxation ii: numerical implementation with 3 and 5 waves. *Numerische Mathematik*, 115(4):647–679, 2010.
8. B. Bramas, P. Helluy, L. Mendoza, and B. Weber. Optimization of a discontinuous Galerkin solver with OpenCL and StarPU. *International Journal on Finite Volumes*, 15(1):1–19, 2020.
9. G. Q. Chen, C. D. Levermore, and T. P. Liu. Hyperbolic conservation laws with stiff relaxation terms and entropy. *Communications on Pure and Applied Mathematics*, 47(6):787–830, 1994.
10. S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.
11. D. Coulette, E. Franck, P. Helluy, M. Mehrenberger, and L. Navoret. *Palindromic Discontinuous Galerkin Method*, pages 171–178. Springer International Publishing, Cham, 2017.
12. D. Coulette, E. Franck, P. Helluy, M. Mehrenberger, and L. Navoret. High-order implicit palindromic Discontinuous Galerkin method for kinetic-relaxation approximation. *Computers & Fluids*, 190:485–502, 2019.
13. C. Courtès, D. Coulette, E. Franck, and L. Navoret. Vectorial kinetic relaxation model with central velocity. application to implicit relaxations schemes. *Communications in Computational Physics*, 27(4), 2020.

14. J.-P. Croisille, R. Khanfir, and G. Chanteur. Numerical simulation of the MHD equations by a kinetic-type method. *Journal of scientific computing*, 10(1):81–92, 1995.
15. R. B. Dahlburg and J. M. Picone. Evolution of the Orszag-Tang vortex system in a compressible medium. i. initial average subsonic flow. *Physics of Fluids B: Plasma Physics*, 1(11), 1989.
16. A. Dedner, F. Kemm, D. Kröner, C.-D. Munz, T. Schnitzer, and M. Wesenberg. Hyperbolic divergence cleaning for the MHD equations. *Journal of Computational Physics*, 175(2):645–673, 2002.
17. P. J. Dellar. Lattice kinetic schemes for magnetohydrodynamics. *Journal of Computational Physics*, 179(1):95–126, 2002.
18. P. J. Dellar. An interpretation and derivation of the lattice Boltzmann method using Strang splitting. *Computers & Mathematics with Applications*, 65(2):129–141, 2013.
19. F. Drui, E. Franck, P. Helluy, and L. Navoret. An analysis of over-relaxation in a kinetic approximation of systems of conservation laws. *Comptes Rendus Mécanique*, 347(3):259–269, 2019.
20. F. Dubois. Equivalent partial differential equations of a lattice Boltzmann scheme. *Computers & Mathematics with Applications*, 55(7):1441–1449, 2008.
21. F. Dubois. Simulation of strong nonlinear waves with vectorial lattice boltzmann schemes. *International Journal of Modern Physics C*, 25(12):1441014, 2014.
22. M. Dumbser and R. Loubère. A simple robust and accurate a posteriori sub-cell finite volume limiter for the Discontinuous Galerkin method on unstructured meshes. *Journal of Computational Physics*, 319:163 – 199, 2016.
23. N. Frapolli, S. S. Chikatamarla, and I. V. Karlin. Entropic lattice Boltzmann model for compressible flows. *Physical Review E*, 92(6):061301, 2015.
24. N. Frapolli, S. S. Chikatamarla, and I. V. Karlin. Theory, analysis, and applications of the entropic lattice Boltzmann model for compressible flows. *Entropy*, 22(3):370, 2020.
25. B. Gaster, L. Howes, D. R. Kaeli, P. Mistry, and D. Schaa. *Heterogeneous computing with OpenCL: revised OpenCL 1.* Newnes, 2012.
26. B. Graille. Approximation of mono-dimensional hyperbolic systems: A lattice Boltzmann scheme as a relaxation method. *Journal of Computational Physics*, 266:74–88, 2014.
27. P. Helluy, T. Strub, M. Massaro, and M. Roberts. Asynchronous OpenCL/MPI numerical simulations of conservation laws. In *Software for Exascale Computing-SPPEXA 2013-2015*, pages 547–565. Springer, 2016.
28. R. Keppens, O. Porth, and C. Xia. Interacting tilt and kink instabilities in repelling current channels. *The Astrophysical Journal*, 795(1):77, 2014.
29. A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih. PyCUDA and PyOpenCL: A scripting-based approach to gpu run-time code generation. *Parallel Computing*, 38(3):157–174, 2012.
30. S. Lankalapalli, J. E. Flaherty, M. S. Shephard, and H. Strauss. An adaptive finite element method for magnetohydrodynamics. *Journal of Computational Physics*, 225(1):363–381, 2007.
31. J. Latt, C. Coreixas, J. Beny, and A. Parmigiani. Efficient supersonic flow simulations using lattice Boltzmann methods based on numerical equilibria. *Philosophical Transactions of the Royal Society A*, 378(2175):20190559, 2020.
32. D. O. Martínez, S. Chen, and W. H. Matthaeus. Lattice Boltzmann magnetohydrodynamics. *Physics of plasmas*, 1(6):1850–1867, 1994.
33. S. A. Orszag and C.-M. Tang. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. *Journal of Fluid Mechanics*, 90(1):129–143, 1979.
34. H. Otomo, B. M. Boghosian, and F. Dubois. Two complementary lattice-Boltzmann-based analyses for nonlinear systems. *Physica A: Statistical Mechanics and its Applications*, 486:1000–1011, 2017.

35. J. M. Picone and R. B. Dahlburg. Evolution of the Orszag-Tang vortex system in a compressible medium. ii. supersonic flow. *Physics of Fluids B: Plasma Physics*, 3(29), 1991.
36. K. G. Powell. An approximate Riemann solver for magnetohydrodynamics (that works in more than one space dimension. Technical Report NASA/CR-194902 ICASE Report No. 94-24, ICASE-NASA Langley, ICASE, NASA Langley Research Center, April 1994.
37. F. Renard, Y. Feng, J.-F. Boussuge, and P. Sagaut. Improved compressible hybrid lattice Boltzmann method on standard lattice for subsonic and supersonic flows. *Computers & Fluids*, page 104867, 2021.
38. R. L. Richard, R. D. Sydora, and M. Ashour-Abdalla. Magnetic reconnection driven by current repulsion. *Physics of Fluids B: Plasma Physics*, 2(3):488–494, 1990.
39. H. Strauss and D. Longcope. An adaptive finite element method for magnetohydrodynamics. *Journal of Computational Physics*, 147(2):318 – 336, 1998.
40. S. Succi. *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford University Press, 2001.
41. M. Torrilhon. Non-uniform convergence of finite volume schemes for Riemann problems of ideal magnetohydrodynamics. *J. Comput. Phys.*, 192(1):73–94, 2003.
42. G. Tóth. The $\nabla \cdot B = 0$ constraint in shock-capturing magnetohydrodynamics codes. *Journal of Computational Physics*, 161(2):605–652, 2000.
43. J. Zhao. Discrete-velocity vector-BGK models based numerical methods for the incompressible Navier-Stokes equations. *Communications in Computational Physics*, 29(2):420–444, 2021.